Network Security Key Generation Algorithm Based on IP Messages

Samara Mohammed Radhi*, Raheem Ogla[2]
[1]Computer Sciences Department, University of Technology, Baghdad, Iraq
[2]Computer Sciences Department, University of Technology, Baghdad, Iraq

Abstract:
Before sensitive information is sent via a communication network, it must first be encrypted using a cryptographic key, which is the single most critical component of this process. An effective cryptographic key possesses both a random sequence and a long period of time as attributes. In this research, the security of cryptographic methods is increased through the (Triple Data Encryption Standard) algorithm in order to secure the network when key generation is taking place. This is accomplished by utilizing a new way to construct a keys-based IP message header structure. In respect to this paper. The information is sent throughout the network through IPsec cryptographic protocols, which provide three levels of protection against attacks and unauthorized access.

Keywords: Cryptography, Decryption, Encryption, key Generator, Triple Data Encryption Standard, Secret key,
DOI: 10.24297/j.cims.2023.5.8

1. Introduction

Information security professionals frequently use a wide variety of different encryption techniques. There are two main types of encryption algorithms: symmetric-key (also known as private-key encryption) and asymmetric-key (also known as public-key encryption). In a cryptosystem, the same key can be used for both encryption and decryption. Symmetric key cryptography is the gold standard and by far the most popular method of encryption. Data is encrypted and decrypted in a cryptosystem utilizing public key encryption, an asymmetric encryption approach that relies on a set of keys [1], [2]. Data owners are understandably concerned about user privacy; as the data they collect may frequently be used to trace back to specific individuals. Data is shielded from prying eyes by using multiple-receiver planning in conjunction with conditional sharing and anonymity. Safeguards are necessary to avoid the loss of sensitive information and the introduction of malicious code. A cloud service provider is responsible for the security of their customer’ s data, which is stored in a massive database on

their behalf. By using the receiver's public key, information can be encrypted such that it is unreadable to anybody other than the recipient [3], [4].

By means of a mathematical algorithm, cryptography transforms a plaintext message into an unreadable cipher text. Cryptography can be used for authentication, which is the process of determining whether or not a given individual is who they claim to be. Cryptography also ensures integrity by making it hard for unauthorized parties to alter the data. This system ensures data privacy by preventing access to it by unauthorized parties. Symmetric crypto algorithms and asymmetric crypto algorithms are the two most common varieties. In a symmetric algorithm, the same key is used for both the encryption and decryption processes. The encryption and decryption processes in an asymmetric algorithm both make use of unique keys. Symmetric crypto algorithms include the well-known DES, AES, and RC6 [5], [7]. This form of encryption is typically employed when sending large amounts of data. The most difficult part of these algorithms is finding a safe means to send the key from sender to recipient. Sender and recipient in an asymmetric cryptosystem do not have access to the same secret key. In a private key crypto scheme, only one of the keys is made public; the other, which only the recipient has access to, is used for encryption. RSA, Diffie-Hellman, and other similar methods are just a few examples. To generate a fixed-length output, or hash value, cryptographic hash functions accept data of any length as input. It's a one-way function, therefore it can only go one way [8]-[10]. Reconstructing the original message from the hash value is really challenging. The primary objective is to boost data transfer safety. To fortify protection, utilize a hybrid cryptosystem.

Combining many distinct cryptographic protocols yields a hybrid cryptosystem [11], [12]. This research presents a hybrid cryptosystem that combines the best features of asymmetric crypto algorithms (Triple DES) and hash functions (MD5).

This paper will follow the following outline: Relevant works are mentioned in Section 2 of this paper. The most basic outline of the suggested approach is presented in Section 3. Experiment procedures are described in detail in Section 4. The performance of the classification algorithm is analyzed and discussed in Section 5. The final results of the paper are reported in Section 6.

2. Related Work

As technology advances, researchers continue to investigate cryptographic key management topics such as key generation, agreeing to or exchanging keys, distributing keys, authenticating keys, and updating keys. This paper focus on key generation for symmetric cryptosystems [13].

Intelligent, active secret key creation was employed as a seed in the evolutionary algorithm (EA) developed by Aboshosha et al. [14]. With the RC4-EA method, the randomness of the seed keys used by the RC4 encryption process is significantly improved. The RC4 encryption algorithm's fundamental defense against being broken is improved by the proposed RC4-EA approach since the secret key is created dynamically and arbitrarily, making it harder to crack. The proposed RC4-EA approach improved both throughput and encryption time, as shown by the results of multiple testing.

In 2016, Munthir et al. [15] presented a method for constructing VPNs using the IPSec protocols. The development approach uses pre-existing packages, the most trustworthy and significant standards, and component-based software engineering (CBSE). The security settings for IPSec are universal, as it functions at the network layer. PreShared Key was used as the authentication method. The suggested technique uses the Linux kernel's IPSec implementation, which delivers AES security, highlighting its architecture and performance. Some issues with IPSec were also addressed and fixed in this work. AES weaknesses have been explained, and a solution based on the Symmetric Random Function Generator (SRFG) has been proposed, by Saha et al. [16]. Introducing randomization into the key generation process of block ciphers is a groundbreaking development. Nonlinearity, resilience, balanced Ness, propagation characteristics, and immunity were some of the criteria the authors used to compare their results to those of the original AES. In comparison to the original AES, the RK-AES performs better in terms of both the avalanche effect and the confusion property.

An enforcement framework for TFC that is based on IPsec was proposed by Kiraly et al. [17]. Differentiating features of this approach include a module developed to impose packet padding, fragmentation, dummy packet generation, and artificially altering the packet forwarding latency, and a TFC header developed to transport data across the IPsec tunnel to enable packet handling on the receiving end. The proposed technique has been implemented in a Linux 2.6 Kernel, as well as the results of experimental testing have been shown to prove its efficacy. In contrast to the methods described above, the proposed method uses the known 3DES with a key size of 128

bits, extracts the information from the message header, and then uses the MD5 hashing algorithm to build a key that is fixed in length.

3. The Proposed System

This section describes the strategies, techniques, and algorithms used to safeguard information transported across public or private networks using the IPsec protection protocol. These secret keys are formed in an inventive fashion, by investing (exploiting) the embedded information in encapsulating the message's content before sending it over the network.

It is common knowledge that the header and the message body (also known as the payload or plain text) are the two most important portions of any networked communication (IP structure). The header describes the message structure and includes the protocol type, source and destination addresses, message length, message size, message identification, time live, kind of service, header checksum, and other information [18], [20] (see Figure (1)). Since each communication has its own unique header, this information is not constant but rather fluctuates among messages.

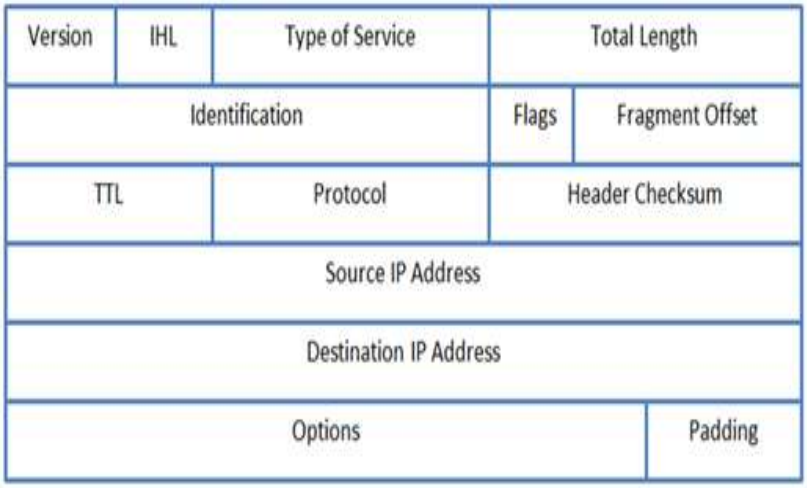

Figure 1. IP header structure

Message headers vary in length and content between sender and receiver. This feature inspired the researcher to use this information to protect messages transmitted over the network with authentication (self-secret keys) and encryption techniques. Developing an IPsec's protection with 3DES algorithm has been implemented to encrypt message content while using DB to protect the system. Because the encrypted data and secret key are kept in the DB in a record, the

sender will request the record ID to send it across the network to the receiver [21]. Figure (2) displays the proposed system's flowchart. When two parties try to interact, the sender prepares the message and then creates a database on the server. The technique extracts each packet's header to generate the initial encryption key. MD5 will be applied to the first encryption key to construct a final secret key, and 3DES will be used to encrypt packets. The encrypted packet and key are delivered to the DB to create a record and assign an ID, and the sender requests that ID from the DB to send it to the recipient across the network. The receiver will decrypt the encrypted packet based on the sender's ID. Below are the steps for the proposed approach:
[1]Header Extraction

Any cryptographic algorithm requires a key to encrypt data. This approach extracts the key from each message's packet header. The header gives considerable information about the message itself, and each message has its own unique information. No other message will hold the same information even if the same environment is produced. This procedure auto-generates a secret key from header data. The suggested approach uses the message header's identification, fragmentation offset, checksum, and sequence number. All the above information is unique to each packet and will be different between messages. Algorithm (1) describes header extraction.

Algorithm (1) Header Extraction

Input: Message.

Output: Header Information.

Step1: *C1 print the message.*
Step2: *Message is split into packets.*
Step3: *The packets enter a header extraction function.*
Step4: *Extract header information.*
Step5: *Reunion the header and the payload.*

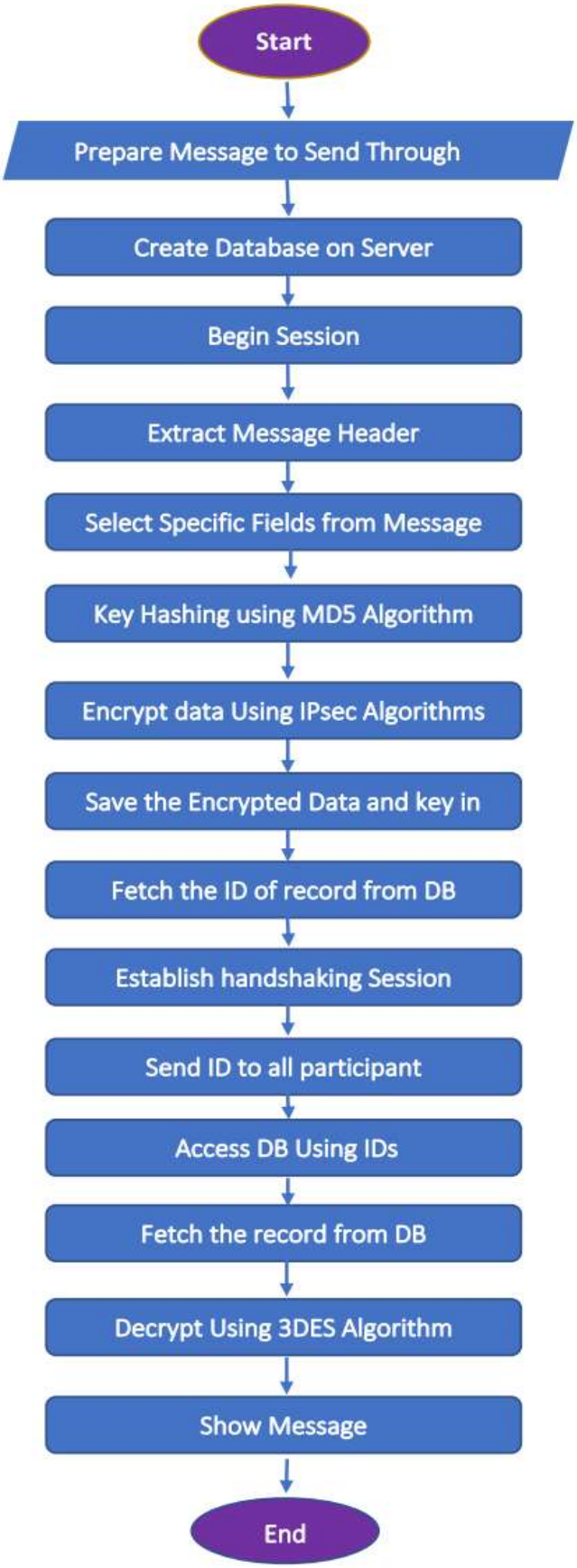

Figure 2. The Proposed System

[2] Key Generation

After the message header has been collected and parsed, the data and the current time will be fed into a secret key generating mechanism. The function's output is hashed with MD5 to obtain a 128-bit fixed secret key. Due to randomization of extracted header information, each packet will have a different encryption secret key than the others. The method for creating keys is outlined in algorithm (2).

Algorithm (2) Key Generation:

Input: Header information.

Output: 128-bit Cryptographic Secret key.

Step1: *The header information will enter the function of key generation.*

Step2: *Choose the current date and time.*

Step3: *XOR the output from step1 and step2.*

Step4: *Perform hashing of the output of Step3 using the MD5 hash algorithm.*

Step5: *The output of step4 is the secret key.*

The output of algorithm (2) can be hashed using the MD5 hashing algorithm to produce a fixed-length encryption key of 128 bits in size, which can then be used to build the final version of the encryption key. MD5 hashing uses a sophisticated mathematical calculation. It converts the initial encryption key into fixed-size blocks and manipulates them. During this time, the algorithm adds a unique value and transforms the result into a signature or hash. The MD5 algorithm steps are exceedingly complex enough that they can't be reversed to generate the original file from the hash. Identical input yields same output.

[3] Encrypting with 3DES

In this step, the secret encryption key is previously constructed based on the packet header (identity, fragmentation offset, checksum, and sequence number) and date and time. MD5 is used to generate a fixed 128-bit hash value. This section uses 3DES, which has three keying choices. First, each key is independent. K1 and K2 are independent, K3=K1 is a double-length key; and K1 = K2 = K3 is the third keying choice [22], [23]. The suggested system makes advantage of the second keying option, which is known as K1=K3. In this setting, the output of algorithm (2) is divided into two parts: the first 64 bits of the 128-bit total will be known as K1,

and the second 64 bits will be referred to as K2. The suggested system uses the second keying option, hence K1=K3.

Since this is the case, the network layer will perform a binary conversion on the message and encrypt each block individually using a single DES and the key K1. Use a single DES and the key K2 to decrypt the result. Encrypt the output with K3 and DES to produce the output is cipher message. Decryption is the opposite of encryption. Users decrypt K3, encrypt K2, then decrypt K1. The DES encryption mechanism is described in detail in its algorithm (3).

Algorithm (3) Encrypting the data using 3DES

Input: The row data, key, and IP table.

Output: Encrypted message using 3DES

Step1: *Convert plaintext to binary and split it to blocks of 64 bits.*

Step2: *Perform the following steps on each block:*

Apply initial permutation using IP table.

Divide the 64-bit (after IP operation) to two parts right and left sides, and each side has 32bits.

Perform the F function

Take the right side and put it in the expand function; expand from 32bits to 48bits.

After the expand function, take the right side (48bits) and XOR with the key (48bits).

Move the 48bits to subsection operation (S-BOXES), where this operation input for each s-box is 6bits and the output is 4bits , so the result is 32bits.

Apply permutation using the P-BOX table

Apply XOR with the left side.

To continue the 15 rounds, twist each side of 32-bit (the right side is now the left side and the left side is now the right side).

In the final round, the XOR merges two 32-bit to 64-bit and put this result to inverse initial permeation (IP-1).

To continue the 15 rounds, twist each side of 32-bit (the right side is now the left side and the left side is now the right side).

In the final round, the XOR merges two 32-bit to 64-bit and put this result to inverse initial permeation (IP-1).

The suggested approach makes use of a 3DES, which encrypts, decrypts, and encrypts (EDE) data in three distinct phases. Using two separate 64-bit keys (K1 and K2), the first message is encrypted using K1, then decrypted using K2, and lastly encrypted again using K3.

[4]Data encryption and key transmission to database

Integrating the DB into the system will boost security by providing additional encryption for transferred data. As soon as the system boots up, it will initiate a connection to the database by sending a request to the open connection instruction code, which creates a secure, encrypted connection between the client and the server. Each record's field name is shown in Table (1).

Table 1. Database records' description

Field name	Field type	Description
Msg_ID	Big integer	Record ID.
Msg_Text	Varchar	The ciphered packet.
Msg_Key	Nvarchar	Encryption key.
Msg_DateTime	Nvarchar	Time of sending the packet.
Msg_UserID	Big integer	Sender ID.
Unique_Note	Nvarchar	Identifications.
Msg_Note	Nvarchar	Notifications.
Enc_Method	Big integer	Encryption method.
Is_image	Integer	Flag which is 0 if the transmitted message is a text, and 1 if the transmitted message is an image.

The sender will request the record ID after the encrypted message and key have been entered into the database and assigned a record. Message encryption ID and sender decryption key can be retrieved from the database. Specifically, this instance uses incremental numbers as the ID.

[5] Network ID transmission

To further improve the system's dependability and security, once the sender obtains the ID, then will just send that ID via the network to the receiver, bypassing the network entirely and not even

sending the encrypted message. Since the ID is a sequential number, hackers and man-in-the-middle attacks won't cause any system security weaknesses.

[6] Get the Encrypted Record

The sender, who is also connected to the database, will give the ID to the receiver, who will then use the get command to request the encrypted message and key from the database based on the ID. The proposed system establishes a number of security procedures that occur within the network layer as data travels from client to server on the network.

[7] Decrypting the message

Since both parties are linked to the DB, the sender will provide the recipient with the ID. The recipient will then use the ID they obtained from the sender to retrieve the encrypted message and key from the database. Then, the recipient uses the secret key extracted from the database to read the original message. The final step is to display the message on the screen. As illustrated in algorithm (4)

Algorithm (4) Decrypting the packet

Input: ID.

Output: Decrypted message on the screen.

Step1: *Retrieve the encrypted message and the key from DB based on ID.*

Step2: *Decrypt the packet using the secret key for the 3DES encryption algorithm.*

Step3: *Print the raw message on the screen.*

4. Implementation Of System

All of the proposed system processes are carried out on a Lenovo laptop equipped with a CORE i7, 2.50 GHz CPU, 64 bit windows, and 6 GB of RAM. The proposed system's information is shown on a graphical interface. Visual Studio C# 2015 IDE and java 8 with NETBEANS IDE 8.2. In addition, the suggested system makes use of SQL server 2014 and PAYTHON 3.6.5.

Figure (3) depicts the primary graphical user interface, which consists of a number of buttons that, when clicked, carry out various processes within the proposed system.

Figure 3. The primary user interface for the proposed system.

IP addresses and port numbers for both sending and receiving will be entered into this interface. When ready, both ends of the connection will hit the connect button to send data to and from the network's shared database management system (DBMS), which is implemented as an instance of Microsoft's SQL Server. A sample of five messages is shown in Table (2) to illustrate how the proposed method might be put into practice.

Table 2. Five messages' samples.

Message ID	Contents
M1	HELLO WORLD
M2	UNIVERSITY OF TECHNOLOGY
M3	MASTER STUDY
M4	ladybug image
M5	zebra image

When the system loads, it creates a connection string and database that all network clients will use. User and server initiate a virtual tunnel connection. Each client has a virtual tunnel connecting to the same server. When the session is established, the DB is built but is empty. User1 sends user2 a message, which will be separated into packets with their own headers. The suggested system extracts header information to produce message encryption keys. Table (3) shows sample messages and extracted headers.

Table 3. Information extracted from the message header.

Message	Extracted Header Information			
Name	Identification	Fragmentation	Checksum	Sequence number

		offset		
M1	53809	16384	0x82ac	436420060
M2	38431	16384	0x914c	738080817
M3	38582	16384	0x4d16	2345359384
M4	57085	16384	0x15db	3584454651
M5	54167	16384	0x82ac	436439750

The proposed system will use the retrieved information and current date and time to generate the initial encryption key. The suggested system will use the MD5 hashing method to construct a fixed-length encryption key from the reduced key (extracted header information and sending date). As indicated in Section 3, the proposed system will use two 3DES keying alternatives. Each encryption key must be 64 bits long to generate two 64-bit secret keys, and the suggested system will use the MD5 hashing technique to construct a 128-bit fixed-length key. As long as 3DES requires 64-bit encryption keys per key, the suggested system will use the first 64 bits of the MD5 hashing method as the first encryption key and the second 64 bits as K2. As shown in Table (4).

Table 4. Messages key encryption

Message Name	K1	K2
M1	XSv1/IM/qfhc/fBUO8ZxOqS+nnOfuK5gboUyw8iAgtiMJ0Mh9Fx9JYPA5I/WJJ67	AeOGezaRod0L2jK1Qc8btQLRkJvprtrKxuWptdxzelMnCCJiXoPw680B58CmYyd3
M2	TPCIV7yzAqTJkfBm8QfcDb6ka/FuKcF1cxT9+Q/s7ASL8yJ84Woo2ls+Dx7p6Tfs	scoGPV5cQDiSXophxAOmFlNcjncxzaDh/1aHtBnSB/E0T9YmYgjJMtYfLKZXSq8t
M3	hNShci+U2Oen2WQaL7YgxoW1nDeiBMRxxX37ocvgvRQj1/OCN8IU5OrItYyNPf68	FASI/KIDGdE9JQ8RkMqPxSqhCa7dU939svwPuavXPoT8TDsJ6j5sA102paJM6+06
M4	eN0+Pzqi2sjJciOnm482tola6o17L5tWfFSqwg/2RiwK4VLtaSm5255hkcNfEOUE	2+T4v17NLMKDwaU04sSTIqBx/yAKwiRM04Gyj5DNPD92AliR8Vyb6YzBHh2d9sCM
M5	QB6QPhMfM40o6llFdR5C+SV/j7NfRvXBSp2Xx5d26RjzK0NT	C7kLDyiVMPalhOU4KjnsNlnLC0deVYfEsc2PxL8JFcvxn8sJtVhtRbLw

W9r/7nsTwCMmilHT 56qPeHwv

After encrypting the message, the sender sends the encrypted message and encryption key to the DB to generate a record and assign an ID, as in Figure (4). The sender will use a set instruction to open a connection string and request a select query where ID = max. This query's get instruction returns the last ciphered message's ID and key.

Results Messages

	Msg_ID	Msg_Text	Msg_Key	Msg_DateTime	Msg_Use...	Unique_K...	Msg_N...	Enc_Meth...	is_image
160	40143	Wfeliq3eyyS00kRq5fcv4==	g7ogUkJqEk1bCzz9+CdUAAdQe91OueBphpwBlOG7lwTnRRLe9K...	8/3/2022 10:41:42 PM	1	53804	NULL	1	0
161	40144	BYm8x1R3EoH3lDjRelHdwjfUPk70D7...	wY8ocaCuxEtXjNzhYxGjYExSESI8FC001ppy1A6+DghI6sOyG6xhbs...	8/3/2022 10:44:48 PM	1	40025	NULL	1	0
162	40145	DtLd4ckA+2OmKcuPVZq6oQ==	/nDNKUZ5NzBtPp6MhmnzFtEgDpUod2Jgx0sEbA/NjHwGH2bJ6oN...	8/3/2022 10:46:59 PM	1	38582	NULL	1	0
163	40146	26w3x+PWBU7Cs0funAV2FuVw7ZfPBi...	374CNfgrCC7zG5rRjm7lkmsSpLSzazocwW8DH+DpKo1dNeYA28e...	8/3/2022 10:49:20 PM	1	57085	NULL	1	1
164	40147	qe8NH9nRrpApCp4wwbjqH6yRa5u6U...	QB6QPhMMM4Go6IFdR5C+5Vij7MRvXB5p2Xa5d26RjzK0NTW9r7...	8/3/2022 10:56:32 PM	1	54187	NULL	1	1

Figure 4. DB records.

The sender will provide the receiver the ID of the record containing the ciphered message and encryption key. Sending simply the ID of the data gives the system a certain amount of strength, particularly in protecting it against middle-man-attacks and hackers, since the communicated data will make no sense to the hacker even if the network is hijacked or eavesdropped. When the receiver receives the ID from the sender, it requests the complete record from the DB based on the ID to retrieve the ciphered message and the encryption key, and then it decrypts the ciphered message using the received secret key.

5. Results And Discussions

To evaluate the proposed system's experimental results, measurement, standards, and analysis must be used. A subset of these metrics is associated with the gendered key, while the rest are more general to the system as a whole. Specific indicators used for the proposed system's analysis and evaluation as discussed below:

Random encryption keys with a sizable key space will make any assault more difficult to pull off. This makes an ordinary encryption algorithm resistant to numerous hacks. In this section, the NIST test, which comprises of 15 advanced randomization tests, was utilized to test the randomness of key vector creation, that relies on message header information.

Several types of randomizations, such as the Discrete Fourier Transform, the Universal randomizer, randomness excursions, etc., were used in the construction of these tests. Two examples of keys generated using data from IP headers are displayed in Table (5).

6. Random Key Analysis

Table 5. NIST Test Suite for suggested key generation method based on M2 and M3 message header information.

Test. No.	Test Name	Result of M2 (256 bits)	Result of M3 (512 bits)
1	Runs	PASS	PASS
2	Frequency of Block	PASS	PASS
3	The Longest Run	PASS	PASS
4	Frequency	PASS	PASS
5	Cumulative Sums	PASS	PASS
6	Rank	PASS	PASS
7	Non-periodic Templates	PASS	PASS
8	Discrete Fourier Transform	PASS	PASS
9	Overlapping	PASS	PASS
10	Serial	PASS	PASS
11	Approximate Entropy	PASS	PASS
12	Random Excursions	not applicated	not applicated
13	Random Excursions Variant	not applicated	not applicated
14	Universal	Dismissed	Dismissed
15	Linear Complexity	PASS	PASS

Mean Squared Error

Mean Squared Error (MSE) evaluates the total squared gaps between the original and encrypted/decrypted message to evaluate de-accuracy noise. According to Equation (1), MSE can be estimated. If MSE is less than 30, the quality of two messages is clear. The MSE for each of the five original, encrypted, and decrypted message samples is shown in Table (6) [23].

$$MSE = \frac{1}{N}\sum_{n=1}^{i=0}\left(\frac{\hat{0}\mathrm{i} - \mathrm{Oi}}{2}\right) \quad \text{.....(1)}$$

Peak Signal to Noise Ratio

Peak Signal to Noise Ratio (PSNR) compares the original and encrypted messages' quality. Increasing the PSNR will result in a more precise de-noising [24]. Equation (2) estimated PSNR. When analyzing multiple encrypted communications, it was discovered that their average PSNR value was 30.862 db, which is indicative of a significant noise level within the encrypted message files, making them more secure. In addition, the decrypted message has an infinite average PSNR value, which proves that the communication was of perfect quality.

$$PSNRdb = 10log \frac{(max)2}{MSE} \quad \text{.....(2)}$$

Information Entropy Measurement

Higher entropy indicates a more secure method against attacks [25],[26]. The average entropy for decrypted M5 messages is 5.7055, whereas the average entropy for corresponding encrypted files is 6.2658. Based on the entropy values, it is clear that the suggested approach increases the entropy of signals, making them more valuable as they are less predictable and hence less vulnerable to assaults.

Table 6. Results of MSE, PSNR, and Information Entropy Measurement for M4 and M5.

Msg ID	MSE	PSNR	Entropy value for encrypted messages	Entropy value for decrypted messages
M4	4.15	43.2	7.2653	7.1815
M5	3.09	32.4	6.2658	5.7055

Time Analysis

In addition to performance, a cryptosystem's execution time is also important. Programming level and compiler effect system execution time. As shown in Table (7), the desired computation time increases linearly as the message size increases, whereas encryption, decryption, and transmission take very little time compared to message size.

Table 7. Results of encryption and decryption time of the proposed algorithm.

Msg Name	File Size	Encryption Time	Decryption Time
M1	178KB	0.013 /s	0.015 /s

M2	226KB	0.016 /s	0.016 /s
M3	512KB	0.026 /s	0.016 /s
M4	525KB	0.031 /s	0.016 /s
M5	622KB	/s	0.19s

Proposal Attacks

The data transferred over the network is the ID of the record at the DB that has the ciphered message in it, and does not reflect the original data, therefore attacks, hacking, and eavesdropping on the network to learn the contents would fail. The database server has multiple layers of protection, which increases the safety of the proposed system. Furthermore, the encryption data is stored in the record, so even if the hacker attempts to decrypt the data, they will not succeed quickly due to the strength of the encryption key. Because this proposed system allows for each communication to have its own unique encryption key, even if a hacker were to eventually figure out the key, by that point the information would be irrelevant and only be able to decrypt a single message.

7. Conclusions

The IPsec algorithm is used to create a secure networking system for data transmission in this paper. The data is transmitted across the network using IPsec cryptographic techniques, which offer three levels of protection against assaults and illegal access. The IPSec algorithm, which guarantees authenticity and encryption, has been used to build and implement a secure network system. Also, the designed cryptography methodology provides rapid, reliable, and powerful security for data transmission, and the proposal has multiple levels of security represented in the encryption key generation process, utilizing the database as a server to store the encryption message, and finally using the ID of the record in the DB that carries the encrypted message and the encryption key to transmit over the network. Since the encryption key is generated automatically using information in the message headers that is different from message to message, the proposal at the sender side will extract the IP header information for each message and particular header fields to apply the MD5 hashing technique in order to produce two keys to be used in the 3DES encryption process, and each message will be encrypted using a unique encryption key. In addition to, the sender will only communicate the ID over the network to the recipient, and the encrypted data and secret key will be stored separately in a record for each message. The receiver will take the ID and use it to retrieve the encrypted message and encryption key from the database, where they will be kept together. The receiver will then use

the encryption key to decrypt the ciphered message, which is also saved in the same record as the original data.

REFERENCES

1. N. M. Alsaidi, A. T. Sadiq, and A. A. Majid, "CQTRU: A Commutative Quaternions Rings Based Public key Cryptosystem," *Engineering and Technology Journal*, vol. 34, no. 6 Part (B) Scientific, 2016.
2. D. D. Salman, R. A. Azeez, and A. M. J. Hossen, "Key generation from multibiometric system using meerkat algorithm," *Eng. Technol. J.,* vol. 38, no. 3, pp. 115–127, 2020.
3. T.W. Khairi, 2022. "Framework For Modeling and Simulation of Secure Cloud Services," *Iraqi Journal of Computers, Communications, Control and Systems Engineering*, vol. 22, no. 1,2022.
4. R. M. Zaki, T. W. A. Khairi, and A. E. Ali, "Secure Data Sharing Based on Linear Congruetial Method in Cloud Computing," in *Lecture Notes in Networks and Systems*, vol. 201, 2021.
5. A. Jan, S. A. Parah, M. Hussan, and B. A. Malik, "Double layer security using crypto-stego techniques: a comprehensive review," *Health and Technology*, vol. 12, no. 1, pp. 9–31, Oct. 2021.
6. N. A. Wahid, A. Ali, B. Esparham, and M. Marwan, "A Comparison of Cryptographic Algorithms: DES, 3DES, AES, RSA and Blowfish for Guessing Attacks Prevention," *Journal of Computer Science Applications and Information Technology*, vol. 3, no. 2, 2018.
7. P. Singh and S. Kumar, "Study & analysis of cryptography algorithms : RSA, AES, DES, T-DES, blowfish," *International Journal of Engineering & Technology*, vol. 7, no. 1.5, p. 221, Dec. 2017.
8. T. K. Hazra, A. Mahato, A. Mandal, and A. K. Chakraborty, "A hybrid cryptosystem of image and text files using blowfish and Diffie-Hellman techniques," in *2017 8th Industrial Automation and Electromechanical Engineering Conference, IEMECON 2017*, 2017
9. K. N. Sreehari and R. Bhakthavatchalu, "Implementation of hybrid cryptosystem using des and MD5," in *Proceedings of the 3rd International Conference on Communication and Electronics Systems, ICCES 2018*, pp. 52–55, Oct. 2018.
10. M. Indu Maheswari and S. Revathy, "Privacy preserving in bigdata using hashing technique with MD5 and DES," *International Journal of Pharmacy and Technology*, vol. 8, no. 2, pp. 12598–12608, Jun. 2016.

11. A. N. Mazher and J. Waleed, "Implementation of Modified GSO Based Magic Cube Keys Generation In Cryptography," *Eastern-European Journal of Enterprise Technologies*, vol. 1, no. 9(109), 2021.
12. A. N. Mazher and J. Waleed, "Retina based glowworm swarm optimization for random cryptographic key generation," *Baghdad Science Journal*, vol. 19, no. 1, 2022.
13. B. Kumar, M. Hussain, and V. Kumar, "BRRC: A Hybrid Approach Using Block Cipher and Stream Cipher," *Progress in Advanced Computing and Intelligent Engineering*, pp. 221–231, 2018.
14. A. Aboshosha, K. A. ElDahshan, E. K. Elsayed, and A. A. Elngar, "EA based dynamic key generation in RC4 ciphering applied to CMS," *International Journal of Network Security*, vol. 17, no. 4, 2015.
15. M. B. Tuieb and K. K. Jabbar, "VPN Development Based on IPSec Protocols Suit," *Mustansiriyah J. Sci.*, vol. 27, no. 1, 2016.
16. R. Saha, G. Geetha, G. Kumar, and T. Kim, "RK-AES: An Improved Version of AES Using a New Key Generation Process with Random Keys," *Security and Communication Networks*, vol. 2018, pp. 1–11, Nov. 2018.
17. F. Hauser, M. Haberle, M. Schmidt, and M. Menth, "P4-IPsec: Site-to-Site and Host-to-Site VPN with IPsec in P4-Based SDN," *IEEE Access*, vol. 8, 2020.
18. G. Xu, "Research on the application of the IPv6 network protocol," in *Journal of Physics: Conference Series*, vol. 2031, no. 1, 2021.
19. R. W. Abd Aljabar and N. F. Hassan, "Encryption VoIP based on Generated Biometric Key for RC4 Algorithm," *Engineering and Technology Journal*, vol. 39, no. 1B, pp. 209–221, Mar. 2021.
20. R. M. Zaki and H. B. A. Wahab, "4G Network Security Algorithms: Overview," *International Journal of Interactive Mobile Technologies*, vol. 15, no. 16, 2021.
21. A. Vuppala, R. S. Roshan, S. Nawaz, and J. V. R. Ravindra, "An Efficient Optimization and Secured Triple Data Encryption Standard Using Enhanced Key Scheduling Algorithm," in *Procedia Computer Science*, vol. 171, pp. 1054–1063, 2020.
22. K. Bhanu Prakash, S. Sengan, and P. Dadheech, "Implementation of New Secure File Transfer Protocol Using Triple-DES and MD5," *International Journal of Advanced Science and Technology*, vol. 29, no. 6, pp. 4156–4170, 2020.
23. H. W. Dhany, F. Izhari, H. Fahmi, M. Tulus, and M. Sutarman, "Encryption and Decryption using Password Based Encryption, MD5, and DES," *Proceedings of the International*

Conference on Public Policy, Social Computing and Development 2017 (ICOPOSDev 2017), 2018.

24. H. M. Fadhil, "Accelerating Concealed ISB Steganography and Triple-DES Encryption using Massive Parallel GPU," *J Appl Sci Res*, vol. 13, no. 3, pp. 17–26, Mar. 2017.
25. L. Kong, H. Pan, X. Li, S. Ma, Q. Xu, and K. Zhou, "An information entropy-based modeling method for the measurement system," *Entropy*, vol. 21, no. 7, 2019.
26. M. S. Croock, "Keyboard Encryption Algorithm Based on Software Engineering Security," *International Journal of Computing and Digital Systems*, vol. 11, no. 1, pp. 1309–1317, Apr. 2022.